(12) EUROPEAN PATENT APPLICATION

(54) **Microprocessor having branch prediction function.**

(57) A microprocessor has a branch prediction function in which an instruction prefetch operation for a branch target instruction responsive to a branch instruction is performed successively to an instruction prefetch operation for the branch instruction and comprises an instruction queue for temporarily storing the prefetched instruction data, an instruction decoder for decoding instruction data supplied thereto, an aligner for supplying a part of the instruction data stored in the instruction queue to the instruction decoder in response to aligning information, a branch hit table including a plurality of buffer areas, each of which stores less significant bit data of address information used for the instruction prefetch operation for the branch target instruction, and aligner control unit responding to the branch instruction being supplied to the instruction decoder to change the content of the aligning information in accordance with data stored in the corresponding buffer area of the branch hit table.
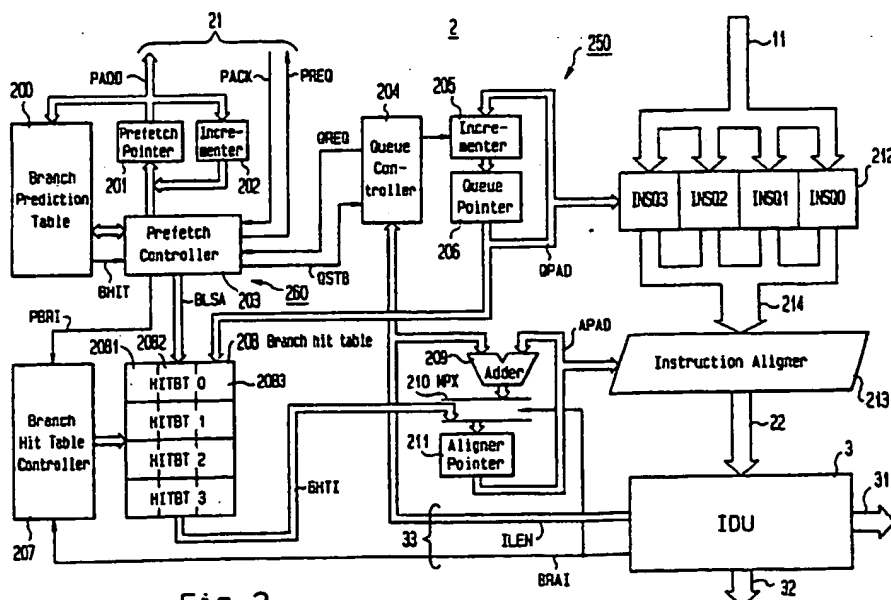
Fig.2

EP 0 458 305 A2

BACKGROUND OF THE INVENTION

The present invention relates to a microprocessor and, more particularly, to a pipelined microprocessor having a branch prediction function.

An instruction prefetch function and a pipelined processing function are widely used to enhance the program execution efficiency of a microprocessor. However, when the microprocessor encounters a unconditional or conditional branch instruction, the execution efficiency thereof are often lowered remarkably. This is because the branch instruction changes the instruction stream to be executed, so that the address to be accessed by the instruction prefetch function is changed with suspending the pipelined processing operation.

In order to solve this drawback, therefore, a branch prediction function is provided to the microprocessor. In this function, a branch prediction table is provided to store in pairs an address of a branch instruction and a branch address designated by the branch instruction. When the branch instruction is prefetched, therefore, a branch prediction hit signal is derived from the table. In response to the hit signal, an instruction stored in the branch address, i.e. branch target instruction, is prefetched in place of prefetching instructions succeeding to the branch instruction. The disturbance in the pipelined processing operation is thus prevented or suppressed.

However, the following problem rises in turn in a recent high performance microprocessor. Specifically, a microprocessor performs in general a data processing operation in word units, so that the instruction prefetch operation is also performed in word units. In a recent high performance microprocessor, the bit length of one word is expanded. Assuming that one word is constructed of 32 bits, instruction data is fetched in 4-byte units per one prefetch operation. The fact that the instruction prefetch operation is performed in 4-byte units means that the memory access for prefetching respective instructions is performed with disregarding the less significant two bits (including the least significant bit) of the address. On the other hand, the byte length of respective instructions including the branch target instruction is not constant, but changes depending on the required data processing operation and/or the addressing mode for operand data. For this reason, the leading byte of each instruction is not always coincident with the word boundary. That is, the leading byte of the branch target instruction is often different from the first byte of four bytes fetched in fact by the prefetch operation for the branch target instruction. Accordingly, such information is required that represents which byte of four bytes fetched in fact

target instruction. Without this information, the decoding operation on the branch target instruction is delayed to disturb the pipelined processing operation.

Moreover, a recent program has been complicated to comply with a higher level processing operation, and in such a program a branch instruction is often written in the branch address. That is, the branch target instruction may be also a branch instruction for requiring a further branch target instruction. In this case, unless there is information for representing which byte of four bytes fetched in fact by the prefetch operation for the further branch target instruction corresponds to the leading byte of the further branch target instruction, the subsequent instruction prefetch operation and the instruction decoding operation are suspended to disturb the pipelined processing operation.

SUMMARY OF THE INVENTION

Therefore, an object of the present invention is to provide a microprocessor having an improved branch prediction function.

Another object of the present invention is to provide a microprocessor in which the disturbance in a pipelined processing operation is prevented or suppressed even when a branch target instruction is also a branch instruction.

A microprocessor according to the present invention has a branch prediction function in which an instruction prefetch operation for a branch target instruction responsive to a branch instruction is performed successively to an instruction prefetch operation for the branch instruction and comprises an instruction queue for temporarily storing the prefetched instruction data, an instruction decoder for decoding instruction data supplied thereto, an aligner for supplying a part of the instruction data stored in the instruction queue to the instruction decoder in response to aligning information, a branch hit table including a plurality of buffer areas, each of which stores less significant bit data of address information used for the instruction prefetch operation for the branch target instruction, and aligner control unit responding to the branch instruction being supplied to the instruction decoder to change the content of the aligning information in accordance with data stored in the corresponding buffer area of the branch hit table.

Since the data stored in the buffer area of the branch hit table represents the position of the leading byte of the branch target instruction within the accessed one word, the aligner can control the position of instruction data to be supplied to the decoder such that the leading byte of the branch target instruction corresponds to the first byte of

Moreover, since the branch hit table includes a plurality of buffer areas, the branch hit occurs successively, the pipelined operation is free from the disturbance.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects, advantages and features of the present invention will be more apparent from the following description taken in conjunction with the accompanying drawings, in which;

Fig. 1 is a block diagram representative of a microprocessor according to an embodiment of the present invention;

Fig. 2 is an internal block diagram representative of an instruction prefetch unit (PFU) shown in Fig. 1;

Fig. 3 is an instruction stream diagram used for explaining an operation of PFU shown in Fig. 2;

Fig. 4 is a timing chart representative of an operation of PFU shown in Fig. 2;

Fig. 5 is an internal block diagram representative of PFU according to another embodiment of the present invention; and

Fig. 6 is a timing chart representative of an operation of PFU shown in Fig 5.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to Fig. 1, a microprocessor 100 according to an embodiment of the present invention includes a bus control unit (BCU) 1 which initiates bus cycles to a system bus 7 coupled to a memory and peripheral I/O units (not shown) to fetch instruction data and read and write operand data. The instruction data thus fetched are supplied through a bus 11 to an instruction prefetch unit (PFU) 2 and temporarily stored therein. PFU 2 and BCU 1 transfer via a control bus 21 therebetween control data including an instruction prefetch request, an acknowledge signal responsive to the request, address instruction for the instruction prefetch operation and so forth. The instruction from PFU 2 is transferred to an instruction decoder unit (IDU) 3 via a bus 22. IDU 3 decodes the instruction supplied thereto and generates data processing information for executing the instruction, operand access information for accessing operand data and decoded instruction information such as the byte length of the instruction. The data processing information is supplied via a bus 32 to an instruction execution unit (EXU) 6, and the operand access information is supplied via a bus 31 to an effective address generator (EAG) 4. The decoded instruction information is returned to PFU 2. EAG 4 calculates and produces an effective address in response to the operand access information 31 and

supplies it to a memory management unit (MMU) 5 via a bus 41. MMU 5 translates the effective address into a real address and then requests an operand access to BCU 1 through a bus 51. MMU 5 further informs IDU 3 of the completion of the address translation through a signal 52. EXU 6 executes the instruction in response to the information from IDU 3 with receiving or supplying operand data from or to BCU 1. The respective units 1 to 6 operate in a pipelined manner.

In the microprocessor 100 of the present embodiment, one word is constructed of 32-bit, i.e., 4-byte. Therefore, the instruction data of 4-byte length are fetched simultaneously by one instruction prefetch operation and then transferred to PFU 2 through the instruction bus 11 of 4-byte length. On the other hand, in a memory storing instruction data and operand data, a unique address is alloted to each one byte data. Accordingly, the fetching of the instruction data of 4-byte length by one prefetch operation means that the memory access is performed by disregarding the contents of the less significant 2 bits (LSB and $LSB+1$) of the memory address.

Turning to Fig. 2, PFU 2 is composed of an instruction queue section 250 and an instruction prefetch and branch prediction control section 260. The section 250 includes an instruction queue 213 having four pockets INSQ0 to INSQ3. Each of the pockets INSQ0-3 has a 4-byte memory capacity. The bus 11 is coupled in common to the queue pockets INSQ0-3. The 4-byte instruction data on the bus 11 is stored in one of the queue pockets INSQ0-3 selected by queue pointer information QPAD of 2-bit from a queue pointer 206. The information QPAD is updated by an incrementer 205 by one and then restored in the pointer 206. The updating timing of the incrementer 205 is controlled by a queue controller 204 in response to a queue strobe signal QSTB from a prefetch controller 203 of the section 260. The queue controller 204 further responds to instruction length information ILEN, which is of 3-bit and representative of the length of the decoded instruction supplied from IDU 3, and detects whether or not the instruction queue 212 is filled with unexecurted instruction data. When the queue 212 is detected to have vacant pocket(s), the queue controller 204 supplies an instruction request signal QREQ to the prefetch controller 203.

The instruction data stored in the queue pockets INSQ0 to INSQ3 are tied up as instruction data of 16-byte length and then supplied to an instruction aligner 213 via a bus 214. The aligner 213 selects four bytes from the 16-byte instruction data and outputs the selected 4-byte instruction data to IDU3. The position of the leading byte of the 4-byte instruction data to be selected is designated by the

aligning pointer information APAD of 4-bit from an aligner pointer 211. Specifically, the byte length of each instruction is not constant, but is changed depending upon the required operation, the number of operand mode and the address mode for each operand data. Therefore, it is required to control the position of the instruction data to be supplied to IDU 3 in accordance wath the byte length of the previous instruction. For example, assuming that the previous instruction has a 3-byte length positioning on the first to third byte of the 16-byte data on the bus 214, the leading byte of the next instruction positions on the fourth byte thereof, so that the data from the fourth byte to the seventh byte require to be supplied to IDU 3. This aligning operation is performed by the instruction aligner 213 in response to the aligning pointer information APAD. The information APAD is added by the adder 209 with the content of the instruction-byte information ILEN and then supplied to a first input node of a multiplexer (MPX) 210 having a second input node supplied with branch hit information BHTI from a branch hit table 208 which will be described later in detail. The output of MPX 210 is rewritten into the aligner pointer 211. MPX 210 responds to a branch decoding signal BRAI from IDU 3. The signal BRAI inform by the high level thereof that IDU 3 is decoding a branch instruction. When the signal BRAI takes the high level, MPX 210 selects the information BHTI. In the case of the low level signal BRAI, MPX 210 selects the output of the adder 209. The information ILEN and the signal BRAI constitute the decoded instruction information 33.

The instruction prefetch and branch prediction section 260 includes a prefetch pointer 201 outputting a prefetch address information PADD for an instruction prefetch operation. The address information PADD is updated by an incrementer 202 by four since 4-byte instruction data are fetched by one prefetch operation. The updated address is then rewritten into the pointer 201. The prefetch address information PADD is further supplied to a branch prediction table 200. This table 200 includes a plurality of storage areas each storing in pairs an address of a branch instruction and branch information under control of the prefetch controller 203. The branch information includes a branch address designated by the branch instruction and the byte length of a branch target instruction stored in the branch address. The prefetch controller 203 responds to the request signal QREQ from the queue controller 204 and supplies a prefetch request signal PREQ to BCU 1 (Fig. 1). BCU 1 transfers the fetched 4-byte instruction onto the bus 11 with returning an acknowledge signal PACK. In response to the signal PACK, the controller 203

controller 204.

When the prefetch address PADD is coincident with the address of the branch instruction stored in the table 200, a branch hit signal BHIT is produced by the table 200 and then supplied to the prefetch controller 203. In response to the hit signal BHIT, the controller 203 reads out the corresponding branch address from the table 200 and sets it into the prefetch pointer 201. When the instruction request signal QREQ is in the active level, the prefetch controller 203 produes the prefetch request PREQ with the branch address. The branch target instruction is thus prefetched by BCU 1 after prefetching the branch instruction and then transferred onto the bus 11 with returning the acknowledge signal PACK. In response to the signal PACK, the controller 203 produces a branch targer hit signal PBRI together with the strobe signal QSTB. The 4-byte instruction data including at least leading byte of the branch target instruction is thus stored in the instruction queue INSQ designated by the pointer information QPAD. The prefetch controller further produces the contents of less significant two bits (LSB and LSB + 1) of the branch address as information BLSA. The branch target hit signal PBRI is supplied to a branch hit table controller 207 controlling a branch hit table 208. This table 208 includes four buffer areas HITBT0 to HITBT3 and stores and outputs information therein and therefrom in a FIFO (First-In First-Out) manner under control of the table controller 207. Each of the buffer areas HITBT0 to HITBT3 has three portions 2081 to 2083, the first portion 2081 of which operates as a valid bit flag set to logic 1 in response to the signal PBRI and reset to logic 0 in response to the signal BRAI and the second portion 2081 of which is supplied with the information BLSA, i.e. the contents of the less significant two bits of the branch address. The third portion 2083 is supplied with the queue pointer information QPAD. The output of the table 2083 is supplied to MPX 210 as branch hit information BHTI to control the aligner pointer 211 and thus the aligning pointer information APAD.

Next, a detailed operation will be described below by way of example of an instruction stream shown in Fig. 3, wherein the first column indicates an instruction; the second column, an address; the third column a mnemonic code; the fourth column, an operation, and the last column, the instruction byte length. In order to facilitate the understanding, assume that the contents of both the pointers 206 and 211 are zero and the prefetch pointer 201 outputs the address A1 of the instruction ①. Also assume that the branch prediction table 200 stores the address A3 of the branch instruction ③, the corresponding branch address A10 in which the

dress A10 of the branch instruction ⑩, and the corresponding branch address A20. Further assume that the leading byte of the instruction ① positioned on the 0-th byte of one word, that of the instruction ⑩ positioning on 2nd byte of one word, and that of the instruction ⑳ positions on the 1st byte of one word.

By the instruction prefetch operation uwing the address A1, the whole byte (3 bytes) of the instruction ① and the first byte of the instruction ② are fetched and then stored in the queue pocket INSQ0, as shown in Fig. 4. The prefetch address PADD is thereafter incremented and the queue pointer information is also updated to "01". The prefetch operation is performed by use of the incremented address, so that the remaining one byte (2nd byte) of the instruction ②, the whole byte (2 bytes) of the instruction ③ and the first byte of the instruction ④ (not shown in Fig. 3) are stored in the queue pocket INSQ1. At this time, the more significant bits of the prefetch address except the least significant two bits thereof are coincident with those of the address A3 of the branch instruction ③ stored in the branch prediction table 200, and therefore the table 200 produces and supplies the branch hit signal BHIT to the prefetch controller 203. The controller thereby reads out the branch address A10 from the table 200 and then sets it to the pointer 201. The prefetch operation using the branch address A10 is thereby initiated, so that the last two bytes of an instruction ⑨ (not shown in Fig. 3) just before the branch target instruction ⑩ and the whole byte of the instruction ⑩ are stored in the queue pocket INSQ2, as shown in Fig. 4. Since the branch target signal PBRI is generated, the least significant two bits of the branch address A10, "10", and the contents of the queue pointer 206, "10", are stored in the first buffer HITBT0 of the branch hit table 208 with the valid flag thereof being set logic 1 by the controller 207 (see Fig. 4). When the prefetch operation using the address A10 is performed, the branch prediction table 200 produces again the branch hit signal BHIT. In response thereto, the prefetch operation using the address A20 is thereafter performed, so that the last one byte of an instruction ⑲ (not shown in Fig. 3) and three bytes of the branch target instruction ⑳ are stored in the queue pocket INSQ3. The least significant two bits of the address A20, "01", and the contents of the queue pointer 206, "11", are written into the buffer HITBT1 along with the valid flag, as shown in Fig. 4.

On the other hand, since the output APAD of the aligner pointer 211 is zero ("000"), the aligner 213 selects and supplies the 4-byte data of the pocket INSQ0 to IDU 3. The instruction ① is thereby decoded by IDU 3. Since the instruction ① is of a 3-byte instruction, IDU 3 recognizes that

the last byte of the selected 4-byte data, i.e. the first byte of the instruction ②, does not belong to the instruction ① , and thus disregards it. The instruction length information ILEN takes a value of 3, "011". The output APAD of the aligner controller 211 is thereby changed to "0011", as shown in Fig. 4. Since a certain instruction such as the instruction ⑳ has a byte length more than 4-byte, IDU 3 produces the instruction length information ILEN at least two times during the decoding operation of one instruction in order to get the whole byte thereof. In the case of the instruction ①, the whole byte thereof is supplied simultaneously to IDU 3, the information ILEN which has produced the second time takes zero ("0000"). The aligning pointer information APAD therefore does not change, so that the aligner 213 continues to output the whole byte (2 bytes) of the instruction ② and the whole byte (2 bytes) of the instruction ③. After completion of decoding the instruction ②, IDU 3 introduces the output of the aligner 213 to initiate the decoding operation of the next instruction. Since the instruction ② is of a 2-byte, data of the instruction ③ are disregarde during this decoding operation. Similarly to the case of the instruction ① , the aligner pointer 211 is controlled to output the information APAD of "0101". The whole byte of the instruction ③, the first byte of the instruction ④ and the last one byte of the instruction ⑨ are thereby outputted through the aligner 213.

Since the instruction ③ is of a 2-byte, the length information ILEN takes a value of "010", so that the aligning pointer data APAD is changed to "0111". However, the instruction ③ is a branch instruction, and hence the signal BRAI changes the high level during the second production of the length data ILEN, as shown in Fig. 4. As a result, the branch hit information BHTI of the buffer HITBT0, "1010", is set into the aligner pointer 211. The whole byte of the instruction ⑩, the last byte of the instruction ⑲ and the first byte of the instruction ⑳ are thereby selected and outputted to IDU 3 through the aligner 213. When IDU 3 starts to decode the instruction ⑩, the signal BRAI changes to the low level, so that the data of the buffer HITBT0 is invalidated by the controller 207. Similarly, the branch hit information of the buffer HITBT1, "1101", is set in the aligner pointer 211 during the decoding operation of the branch instruction ⑩. The three bytes of the instruction ⑳ and a meaningless one byte data are thus outputted through the aligner 213. When IDU 3 starts to decode the instruction ⑳ , it produces the length information ILEN of a value of 3 ("011") to change the aligning pointer data 211 to "0000".

During the decoding operation described above, the instruction prefetch operation proceeds, so that the remaining four bytes of the instruction

㉚ are stored in the queue pocket INSQ0, as shown in Fig. 4. The whole byte (3 bytes) of the instruction 21 and the first byte of the instruction ㉒ are stored in the pocket INSQ1. The remaining two bytes of the instruction ㉒ and two bytes of a next instruction ㉓ (not shown in Fig. 3) are stored in the pocket INSQ1.

Since content of the aligner pointer 211 changes again to "0000", the remaining four bytes of the instruction ㉔ are supplied to IDU 3. The length data ILEN of "1000" is thereby produced to change the content of the pointer 211 to "0100". The instruction ㉑ thus starts to be decoded.

As described above, even when the branch hit state occurs successively and each of the branch target instructions has the leading byte not corresponding to the word boundary, the decoding operation of IDU 3 is never suspended, resulting in no distarbance in the pipelined processing operation. Since the branch hit table 208 has four buffers HITBT0-3, the microprocessor 100 supports four times occurrence of the branch hit state.

Referring to Fig. 5, there is shown an internal block diagram of PFU according to another embodiment of the present invention, in which the same constituents as those shown in Fig. 2 are denoted by the sane reference numerals to omit the further description thereof. This embodiment is constructed to perform a branch prediction operation of conditional branch instructions. The prefetch controller 203 includes a set of address registers 2031 for storing an address which is used when a branch is not taked due to the fact that the branch condition designated by the conditional branch instruction is not satisfied. Whether or not the branch condition is satisfied is informed by IDU 3 through the signals VCAN and UCAN. The signal VCAN takes the active high level when the branch condition is not satisfied, i.e. when the branch is decided to be not taken, whereas the signal UCAN takes the active high level when the branch condition is satisfied. These signals VCAN and UCAN are supplied to the branch hit table controller 207. The signal VCAN is further supplied to the pointers 206 and 211 and the instruction queue 212 to reset all the contents thereof and further to the queue controller 204 and the prefetch controller 203.

An operation will be described below with reference to Figs. 5 and 6 by way of example of the instruction stream shown in Fig. 3. However, note that the instruction ㉚ is modified from the unconditional branch instruction to a conditional branch instruction ㉚. The instruction ③ is of an unconditional branch type similarly to the first embodiment.

In the case where the branch condition designated by the conditional branch instruction ㉚ is

substantial same operations shown in Fig. 4 are obtained.

On the other hand, if the branch condition of the conditional branch instruction ㉚ is not satified, the signal VCAN is generated at the end of the decoding operation of the instruction ㉚. Accordingly, the pointers 206 and 211 are reset to zero and the instruction queue 212 is cleared as shown in Fig. 5. The queue controller 204 responds to the signal VCAN to supply IDU 3 with a signal QEMP representing that all the pockets INSQ0-3 of the instruction queue 212 are empty. The operation of IDU 3 is thereby suspended. The prefetch controller 203 stores in the address register 2031 the incremented address of the branch address A10. The generation of the signal VCAN indicates that the microprocessor requires to execution instructions subsequent to the conditional branch instruction ㉚. Therefore, the controller 203 sets the content of the register 2031 to the prefetch pointer 201 and issues the instruction prefetch request. Thus, the instruction ⑪ subsequent to the instruction ㉚ is stored in the queue pocket INSQ0 and then supplied to IDU 3 through the aligner 213, as shown in Fig. 6.

It is apparent that the present invention is not limited to the above embodiments but may be modified and changed without departing from the scope and spirit of the invention.

Claims

1. A microprocessor having a branch prediction function in which an instruction prefetch operation for a branch target instruction responsive to a branch information is performed successively to an instruction prefetch operation for said branch instruction, first buffer means for temporarily storing the prefetched instruction data, decoder means for decoding instruction data supplied thereto, aligner means coupled between said first buffer means and said decoder means for supplying a part of the instruction data stored in said first buffer means to said decoder means in response to aligning information, second buffer means including a plurality of areas, each of said areas storing less significant bit data of address information used for said instruction prefetch operation for said branch target instruction, and means responsive to said branch instruction being supplied to said decoder means for changing a content of said aligning information in accordance with data stored in the corresponding area of said second buffer means.
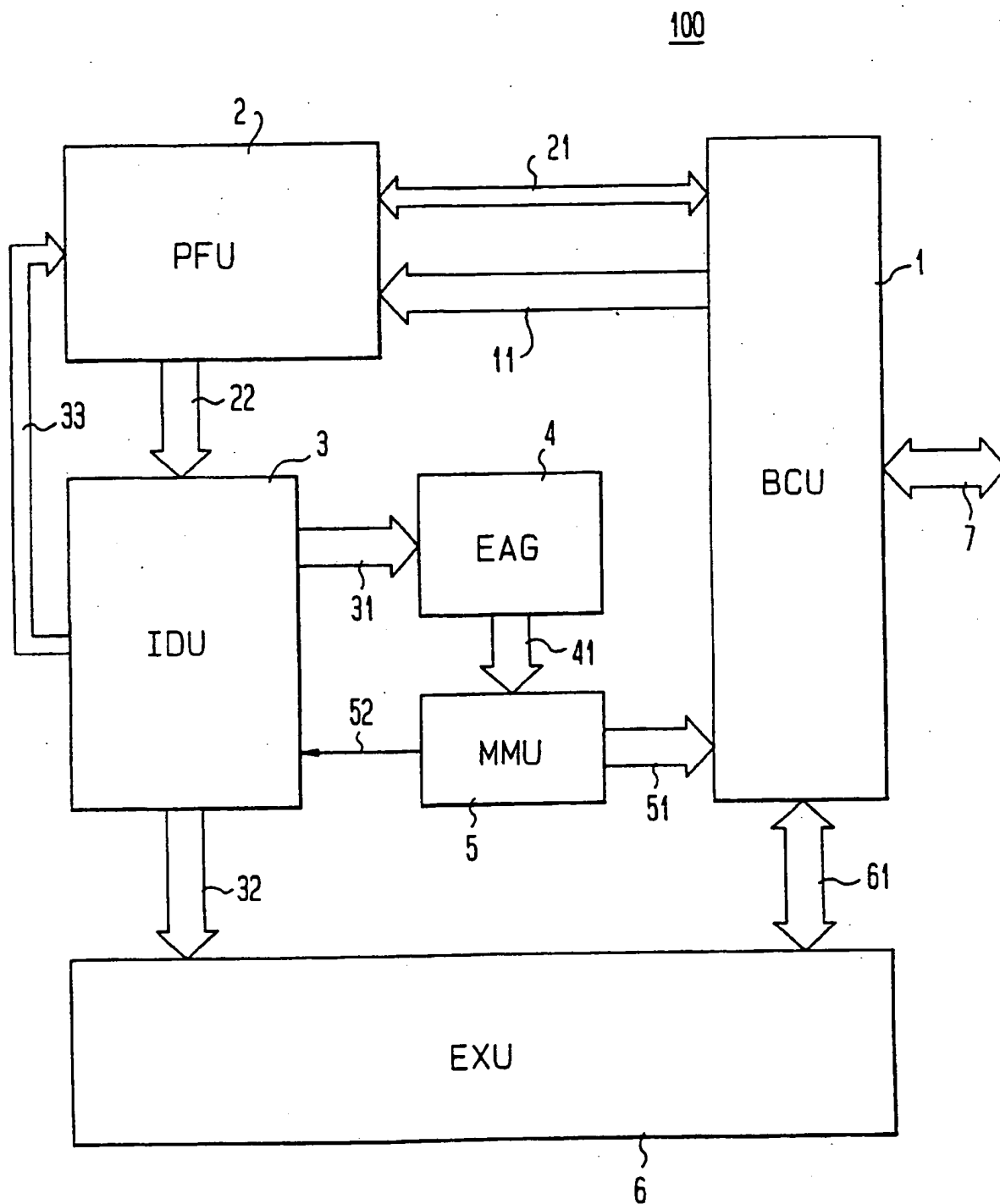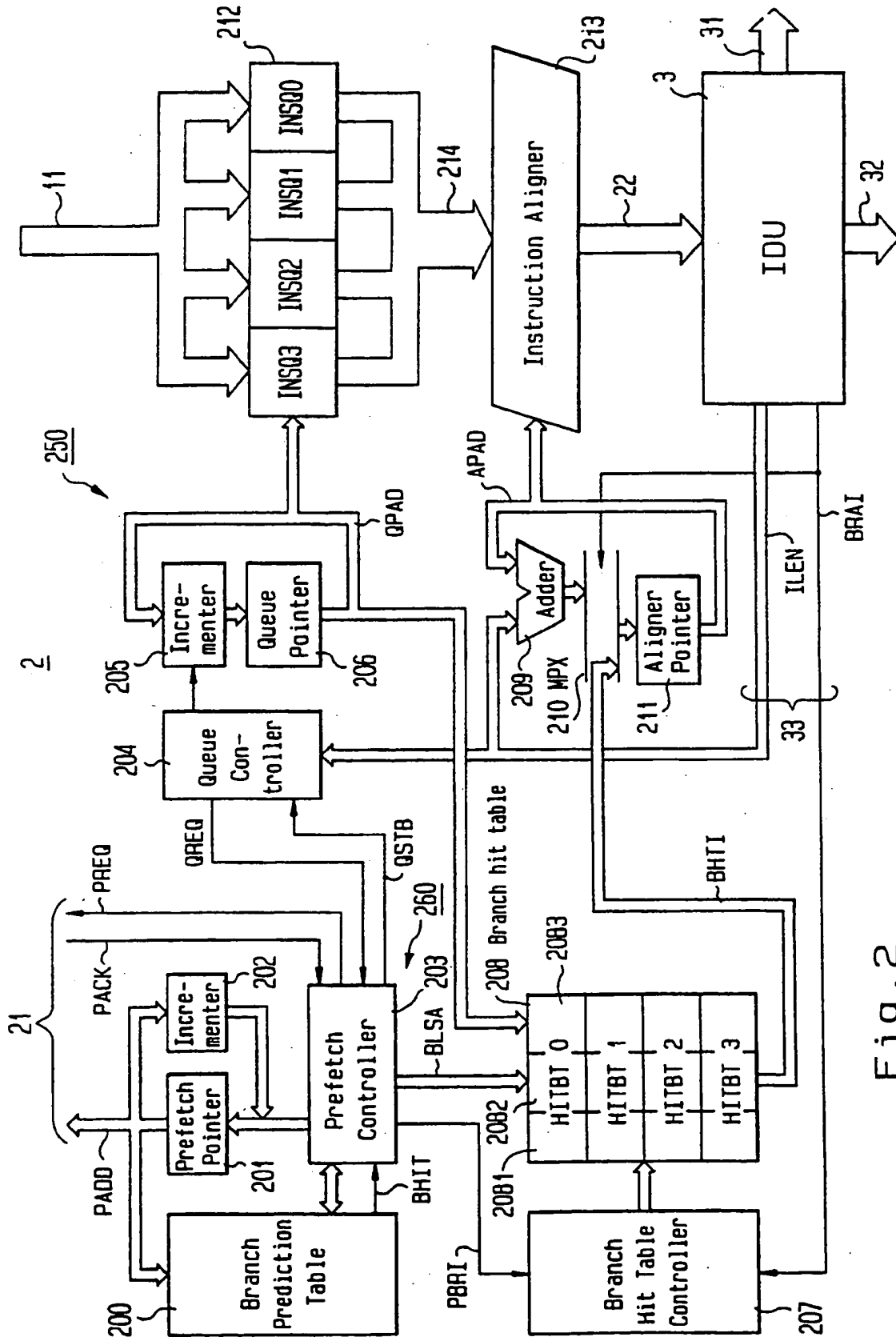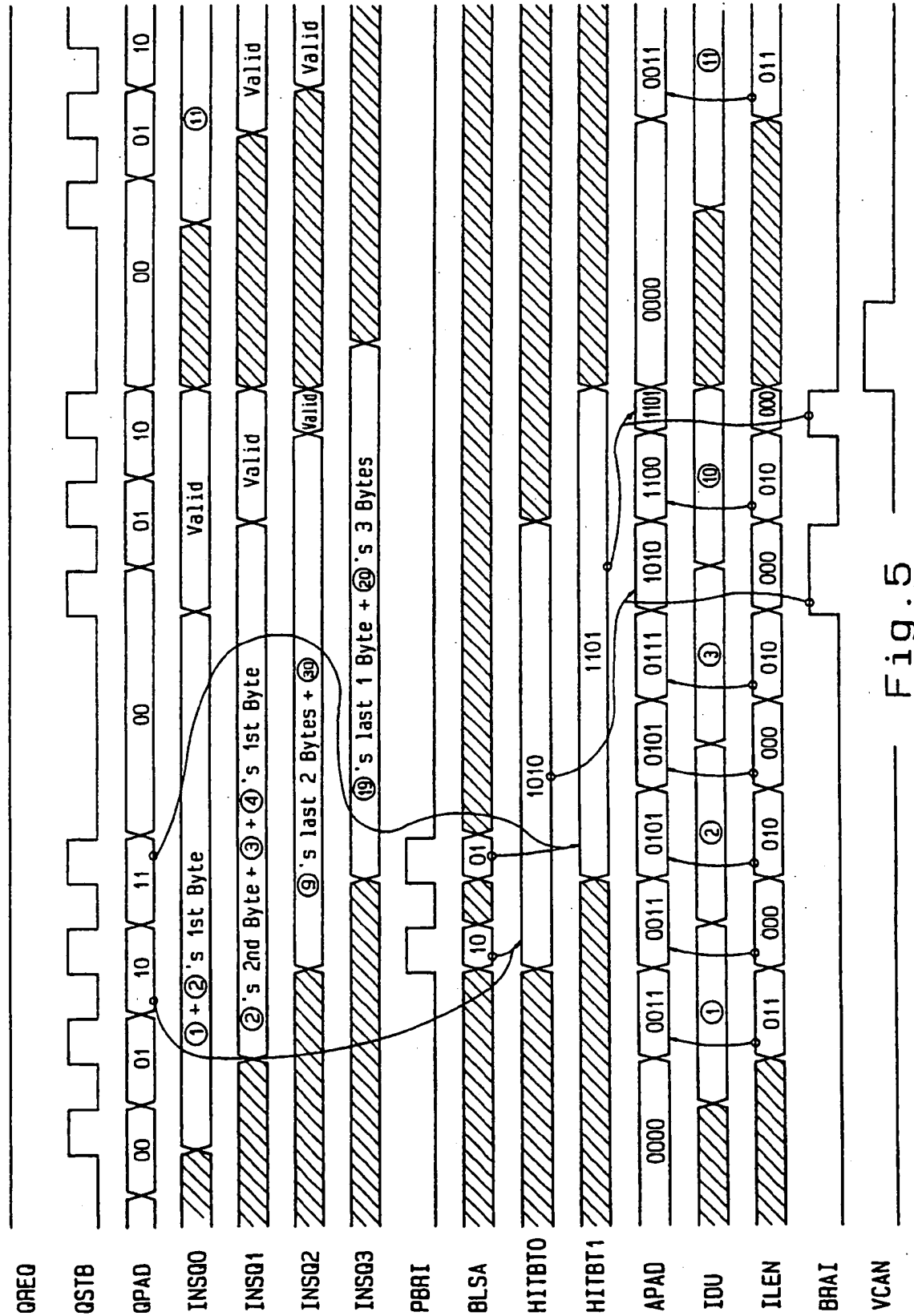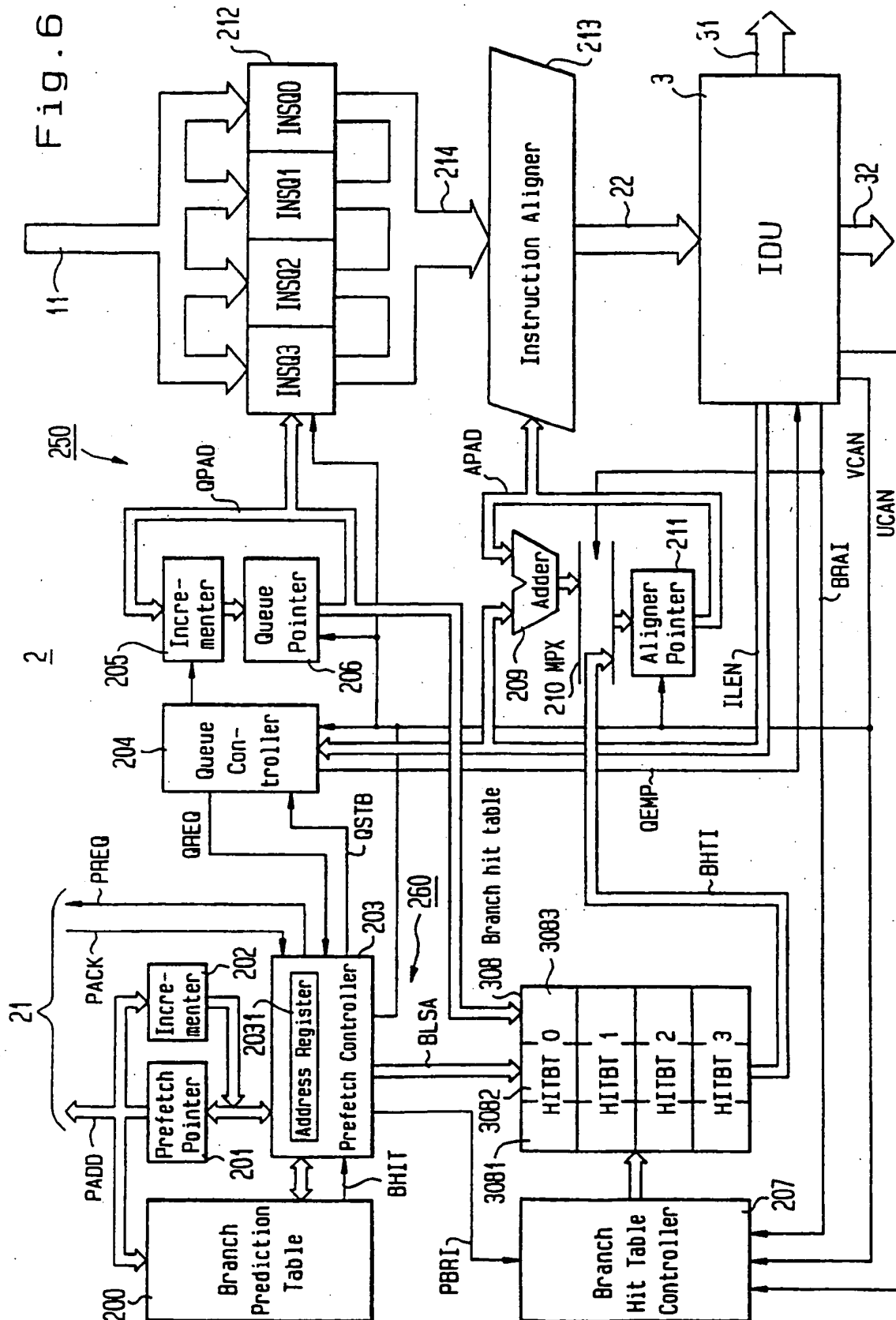
Fig.1

Fig.2

Instruction	Address	Mnemonic	Operation	Byte Length
①	A1	MOV.R1, R2	(R1 — R2)	3
②	A2	INC. R3	(R3 + 1)	2
③	A3	Br. A1	(Branch to A1)	2
		.		
		.		
		.		
⑩	A10	Br. A2	(Branch to A2)	2
		.		
		.		
		.		
⑳	A20	MOV.#0123H, R4	(0123H — R4)	7
㉑	A21	ADD. R4, R5	(R4 + R5 —R5)	3
㉒	A22	SUB. R2, R5	(R5 - R2 —R5)	3
		.		
		.		
		.		

Fig.3

Fig.4

QREQ

QSTB

QPAD   00   01   10   11   00   01   10   10   00   01   10

INSQ0   ① + ②'s 1st Byte   Valid   ①

INSQ1   ②'s 2nd Byte + ③ + ④'s 1st Byte   Valid   Valid

INSQ2   ⑨'s last 2 Bytes + ⑳   Valid   Valid

INSQ3   ⑲'s last 1 Byte + ⑳'s 3 Bytes

PBRI

BLSA   10   01

HITBT0   1010

HITBT1   1101

APAD   0000   0011   0101   0101   0111   1010   1100   1110   0000   0011

IOU   ①   ②   ③   ⑩   ⑪

ILEN   011   000   010   010   000   010   000   011

BRAI

VCAN

Fig. 5

Fig.6

(54) Microprocessor having branch prediction function.

(57) A microprocessor has a branch prediction func-
tion in which an instruction prefetch operation for a
branch target instruction responsive to a branch in-
struction is performed successively to an instruction
prefetch operation for the branch instruction and
comprises an instruction queue for temporarily
storing the prefetched instruction data, an instruction
decoder for decoding instruction data supplied
thereto, an aligner for supplying a part of the in-
struction data stored in the instruction queue to the
instruction decoder in response to aligning informa-
tion, a branch hit table including a plurality of buffer
areas, each of which stores less significant bit data
of address information used for the instruction
prefetch operation for the branch target instruction,
and aligner control unit responding to the branch
instruction being supplied to the instruction decoder
to change the content of the aligning information in
accordance with data stored in the corresponding
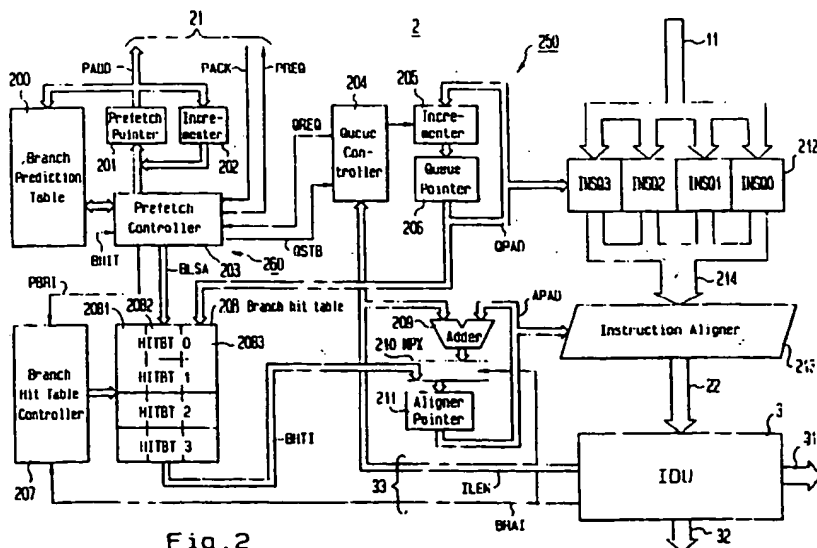buffer area of the branch hit table.

Fig.2

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
A	EP-A-0 135 844 (HITACHI, LTD.) 3 April 1985 * page 6, line 20 - page 11, line 11; figure 4 *	1	G06F9/38
A	EP-A-0 109 655 (NEC CORP.) 30 May 1984 * page 3, line 10 - page 3, line 25 * * page 13, line 11 - page 18, line 1 *	1	

TECHNICAL FIELDS SEARCHED (Int. Cl.5)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	19 MARCH 1993	DASKALAKIS T.

EPO FORM 1503 03.82 (P0401)